Manuel Marroquin
Chips Challenge
10/17/2018

Report

My final design was based on the columbus game in terms of the player movement and map creation. I realized that because I would need 2 levels the original design would not suffice. I opted to implement the strategy pattern for creating levels. As such I created a LevelMap class which took care of building a level from scratch by calling the interface called Level. Level had 2 different “strategies”, one for each Level. LevelMap determined which type of Level to create using a currentLevel counter. Each of these levels had a unique design with keys, gates, portals, and/or monsters all of which were their own classes which could be easily instantiated and reused in future levels.

The main animation, win/lose conditions, and game mechanics were controlled by the overarching class called ChipsChallenge. In this class I created an animation timer and upon each tick checked if any keys were picked up, if any monster collisions occurred, and if the player had reached the portal. If these conditions did occur they were handled accordingly in this class. Upon reaching the next level the LevelMap class was called again to build the next level with the use of a currentLevel counter.

The first level was simple, 4 gates 4 keys, the player must get them in a sequential order to unlock the gate to the portal. The second level is more complicated. The player must progress through 3 unique rooms. Each has more than 1 key for the gate in that room. Only one key is the correct one, the others are rigged and will kill the player. Upon exiting all rooms the player will be in an open area where they will have to make a mad dash to the portal which is in the middle. In this are a player can find 7 moving monster guarding the portal.

If I could change anything about my design it would be the way I handle key pickups. Currently once the key is picked up I hardcoded my programm to make the key disappear and flip a flag to unlock the gate to which it belongs to. In the main class, even though the key is picked up it still checks if the player is on the key space, this is ultimately inefficient because the key is no longer useful after pickup. As such I would have liked to isolate the handling of the key pickup inside the key class instead of hardcoding it into the main class and making it more jumbled. Overall I found that isolating the level creation and creating classes for all of the game components was very useful when creating new levels. New levels could be created in a fraction of the original time once the level component classes were made. Another thing I would have liked to add was more methods for creating walls, all the walls I built had to be created one by one, it would have been nice to create a straight wall given start and end points, or blocks at the call of a function.

Manuel Marroquin
Chips Challenge
10/17/2018

UML Diagram Changes

- Classes for Key, Gate, Portal, and Monster were made.
- More than one level was now created
- The building of each level was Isolated to another class
- Strategy was used to create 2 different levels
- Observer pattern added to control game functions for keys, gates and portals.

Manuel Marroquin
Chips Challenge
10/17/2018

Patterns

Pattern: Observer & Observable	
Class Name	**Role in Pattern**
Player	Observed by Portal and Key
Key	Observers Player and Observed by Gate
Gate	Observes Key
Portal	Observes Player
Purpose: When the player moves the Portal and Keys can check if the player is on top of them and an action can be taken if they are. If the key is picked up then the Key notifies the gate to unlock (or not be solid anymore) to let the player through.	

Pattern: Strategy	
Class Name	**Role in Pattern**
LevelMap	Calls the Level interface to create new level design
Level	The interface which implements strategies for map building
Level01/Level02	“Strategies”
Purpose: We had multiple levels both of which are different, using this pattern allowed me to isolate the parts that change and be able to switch out and create new levels easily.	

Pattern: Facade?	
Class Name	**Role in Pattern**
ChipsChallenge	Client
MapLevel	Facade
Purpose: I isolated ALL of the level creation to MapLevel which allowed the ChipsChallenge class to easily access all of the game elements through the MapLevel class. MapLevel had a list of all monsters, gates, keys, etc. which was used by the main program through this common class that connected every other class.	

Manuel Marroquin
Chips Challenge
10/17/2018